# Review & Perspective for Distance Based Clustering of Vehicle Trajectories

Philippe Besse, Brendan Guillouet, Jean-Michel Loubes, François Royer

# Review & Perspective for Distance Based Clustering of Vehicle Trajectories

Philippe C. Besse, Brendan Guillouet, Jean-Michel Loubes, and François Royer,

***Abstract*—In this paper we tackle the issue of clustering trajectories of geolocalized observations based on distance between trajectories. We first provide a comprehensive review of the different distances used in the literature to compare trajectories. Then based on the limitations of these methods, we introduce a new distance: *Symmetrized Segment-Path Distance (SSPD)*. We compare this new distance to the others according to their corresponding clustering results obtained using both the *hierarchical clustering* and *affinity propagation* methods. We finally present a python package : *trajectory distance*, which contains the methods for calculating the *SSPD distance* and the other distances reviewed in this paper.**

***Index Terms*—Trajectory clustering**

## I. Introduction

A TRAJECTORY is a set of positional information for a moving object, ordered by time. This kind of multidimensional data is prevalent in many fields and applications, for example, for understanding migration patterns through studying trajectories of animals, predicting meteorology with hurricane data, improving athletes performance, etc. Our study concentrates on vehicle trajectories within a road network. The growing use of GPS receivers and WIFI-embedded mobile devices equipped with hardware for storing data enables the collection of an enormous amount of data that can be used to extract relevant information in order, for instance, to find the optimal path to go from point A to point B, detect abnormal behavior, optimize the traffic flow in a city, predict the next location or final destination of a moving object *etc.* This aims actually to build from the data the different features that characterize the different daily movement of the vehicles on the road network. For this, we consider clustering methods for trajectories. Clustering techniques aim to regroup similar trajectories together into groups that are different from one another. The complexity of trajectory makes this a challenging task as objects can move along many different paths in a given area, moreover the road network of a highway does not have the same complexity as that of a city, and finally the road network in a city differs between the suburbs and downtown. In addition, the speed of an object varies between regions, and between paths taken within a single region. Even within the same path the speed depends on exogenous variables such as the time of day, or whether it is a weekday or the weekend.

Several methods can be used to cluster trajectories. We will focus on distance based trajectory clustering but other specific methodologies have been investigated. Dealing with the functional properties of trajectories considered as continuous function of time Gaffney (2009, [1]), Vasquez *et al.* (2004[2]), Hu *et al.* (2006[3]) successfully apply trajectory clustering methods on video-stream trajectories. Gariel et al. [4] also use the continuous definition of trajectory to re-sample the trajectories and obtain time series of equal length. A principal components analysis is then applied on these new trajectories to obtain principal components and finally cluster them. This method is applied to airplane routes. All these methods take into account both the spatial and the temporal aspect of the trajectory, they are not adapted to the vehicle trajectories constrained to a road network whose time progression is very irregular. Rinzivillo *et al.* (2008,[5]) and Kim *et al.*[6]) propose density-based methods. Both require the definition of a density parameter and a minimum cluster size which implies an extensive knowledge of the studied area or a precise question to obtain good results. Hence, these methods are hard to automatically adapt from one dataset to another. Finally Lee *et al.* (2007,[7]) and Wu *et al.* (2014[8]) propose to use clustering methods on trajectory line segments to enable the detection of important areas of flow, though this does not consider the trajectory as a whole path. Our main objective here is to detect the main path and traffic flow which can later be used to study the different behaviors along these paths.

In this context, the goal of this work is to construct, in a data driven way, a collection of trajectories that model the behaviors of car drivers. These models are learned from a data set of car locations. In this work we focus on clustering trajectories having similar paths. This clustering is based on the comparison between trajectory objects, and as such a new definition of distance between the studied trajectory objects is required.

A large amount of work has been done to give new definitions of trajectory distance. Tiakas *et al.*(2009[9]) , Rossi *et al.* (2012[10]), Han *et al.*(2015[11]) or Hwang *et al.*(2005[12]) propose road network based distances. They assume that the trajectories studied are perfectly mapped on the road network. However, this task is strongly dependent on the precision of the GPS device. When the time interval between two GPS locations is significant, several paths on the graph are possible between locations, especially when the network is dense. Moreover it requires the knowledge of the road network. Here, we focus on entirely data driven methods without any *a priori* information. Several methods have been used to cluster data set of trajectories. Clustering methods using Euclidean distance lead to inaccurate results mainly because trajectories have different lengths. Hence, several methods based on warping distance have been defined , Berndt (1994[13]), Vlachos *et al.* (2002[14]), Chen *et al.* (2004[15]), and Chen *et al.* (2005

[16]). These methods reorganize the time index of trajectories to obtain a perfect match between them. Another approach is to focus on the geometry of the trajectories, in particular their shape. Shape distances like Hausdorff and Fréchet distances can be adapted to trajectories but fail to compare them as a single entity. Lin *et al.* (2005[17]) proposed a method based exclusively on the shape of the trajectory but at high computational cost.

In section II the papers definitions, notations and problem statement are introduced. In section III several distances on trajectory are studied and compared. A new distance will be presented in section IV: the Symetrized Segment-Path Distance (SSPD). SSPD is a shape-based distance that does not take into account the time index of the trajectory. It compares trajectories as a whole, and is less affected by incidental variation between trajectories. It also takes into account the total length, the variation and the physical distance between two trajectories. For all these different distances, we obtain different clusterings. So we can compare the distance on these results. The choice of clustering used is detailed section V. The SSPD and the other studied distances were implemented in a python package, *trajectory_distance* available on github. The presentation of this package and the experimental evaluation of these distances with the chosen clustering techniques on some trajectory sets are analyzed in section VI.

## II. Model for trajectory clustering

### A. Trajectory

A continuous trajectory is a function which gives the location of a moving object as a continuous function of time. In our case we will only consider discrete trajectories defined here after.

**Definition 1.** *A trajectory $T$ is defined as*
$T : ((p_1, t_1), \dots, (p_n, t_n))$,
*where $p_k \in \mathbb{R}^2, t_k \in \mathbb{R}\ \forall k \in [1 \dots n],\ \forall n \in \mathbb{N}$ and $n$ is the length of the trajectory $T$.*

The exact locations between time $t_i$ and $t_{i+1}$ are unknown. When these locations are required, a piece wise linear representation is used between each successive location $p_i$ and $p_{i+1}$ resulting in a line segment $s_i$ between these two points. This new representation is called a piece wise linear trajectory. In this representation, no assumption is made about time indexing of segment $s_i$.

**Definition 2.** *A piece wise linear trajectory is defined as $T_{pl}$ : $((s_1), \dots, (s_{n-1}))$ , where $s_k \in \mathbb{R}^4$ and $n_{pl}$ is the length of the trajectory.*

The length of the trajectory $n_{pl}$ is the sum of the lengths of all segments that compose it : $n_{pl} = \sum_{i \in [1 \dots n-1]} \|p_i p_{i+1}\|_2$.

The notation used in this paper are summarized in Table I.

### B. Distance

There are many ways to define how close two objects are from one another. Beyond the notion of mathematical distance, many functions can be used to qualify this dissimilarity. The terminology used in literature to define them is not completely standardized. Therefore we will use the definition established in Deza *et al.* (2009[18]) as a reference.

**Definition 3.** *Let $\mathcal{T}$ be a set of trajectories. A function $d : \mathcal{T} \times \mathcal{T} \mapsto \mathcal{R}$ is called a $dissimilarity$ on $\mathcal{T}$ if for all $T^1, T^2 \in \mathcal{T}$ :*

- $d(T^1, T^2) \geq 0$
- $d(T^1, T^2) = d(T^2, T^1)$
- $d(T^1, T^1) = 0$

*If all of these conditions are satisfied and $d(T^1, T^2) = 0 \implies T^1 = T^2$ $d$ is considered to be a $symmetric$. If the triangle inequality is also satisfied, $d$ is called a $metric$. These notations are summarized in Table II.*

X indicates the required properties for each distances, while $*$ indicates properties that are automatically satisfied (by the presence of the other required properties for the metric).

### C. Desired properties of clustering and distances

Our aim is to regroup trajectories sharing similar behaviour. We want that trajectories in the same cluster, take similar paths. Hence our goal is to define a clustering method that will regroup trajectories

- with similar shape and length
- which are physically close to each other
- which are similar as a whole with more than just similar sub-parts
- all of these properties should be considered without regard to their time indexing

Moreover we want to design a very general procedure which is able to treat all trajectories data, without prior knowledge of the particular geographical location where they are collected. To obtain such clustering, the goal of this work is to find a distance that respects such properties and to succeed in extracting these features. Actually, the desired distance should have the following properties,

- it compares trajectories as a whole
- the compared trajectories can be of different lengths,
- the time indexing can be very different from one trajectory to another
- the trajectories can have similar shapes but can be physically far from each other and vice versa

TABLE I: Notation

| | |
|---|---|
| $\mathcal{T}$ | The set of trajectories |
| $T^i$ | The $i^{th}$ trajectory of set $\mathcal{T}$ |
| $T^i_{pl}$ | The piece wise linear representation of $T^i$ |
| $n^i$ | Length of trajectory $T^i$ |
| $n^i_{pl}$ | Length of the $T^i_{pl}$ |
| $p^i_k$ | The $k^{th}$ location of $T^i$ |
| $p^i_{pl}$ | The set of continuous points that compose $T^i_{pl}$ |
| $s^i_k$ | The line segment between $p^i_j$ and $p^i_{k+1}$ |
| $t^i_k$ | The time index of location $p^i_k$ |
| $\|p_k p_l\|_2$ | The Euclidean distance between $p_k$ and $p_l$ |

TABLE II: Metric Definition

| Property \ Metric Name | | *dissimilarity* | *symmetric* | *metric* |
|---|---|---|---|---|
| Non-Negativity | $D(T^1,T^2) \geq 0$ | X | X | $*$ |
| Symmetry | $D(T^1,T^2) = D(T^2,T^1)$ | X | X | X |
| Reflexivity | $D(T^1,T^1) = 0$ | X | $*$ | $*$ |
| Triangle Inequality | $D(T^1,T^3) \leq D(T^1,T^2) + D(T^2,T^3)$ | | | X |
| Identity of indiscernible | $D(T^1,T^2) = 0 \implies T^1 = T^2$ | | X | X |

- extra parameters should not be required.

## III. Distance on trajectories: a review

Three main kind of distances have been introduced in the literature. The first uses the underlying road network, **Network-Constrained Distance**. These distances will not be detailed in this paper. They assume that the road network is known and that trajectory data are perfectly mapped on it. Distances that do not use the underlying road network can also be classified into two categories: those who only compare the shape of the trajectory, **Shape-Based Distance** and those who take into account the temporal dimension, **Warping based Distance**.

Performance of clustering algorithms using these distances will be compared in section II, as well as their computation cost and their metric properties.

### A. Warping based Distance

Euclidean distance, Manhattan distance or other $L^p$-norm distances are the most obvious and the most often used distances. They compare discrete objects of the same length. They can be used to look for common sub-trajectories of a given length but they cannot be used to compare entire trajectories. Moreover, these distances will compare locations with common indexes one by one. At a given index $i$, location $p_i^1$ of trajectory $T^1$ will be compared only to location $p_i^2$ of trajectory $T^2$. However, these locations can be strongly different according to the speeds of the trajectories. Hence, it makes no sense to compare them without taking this into account. This problem is also common in time series analysis and not restricted to trajectory analysis.

Warping distance aims to solve this problem. To accomplish this, they enable matching locations from different trajectories with different indexes. Then, they find an optimal alignment between two trajectories, according to a given cost $\delta$ between matched location. Several warping based distances have been defined. $DTW$ (Berndt *et al.*, (1994 [13])) and later $LCSS$ (Vlachos *et al.*, 2002[14]), $EDR$ (Chen *et al.*, 2005[16]) and $ERP$ (Chen *et al.*, 2005[15]). These distances are defined the same way, but they use different cost functions.

In order to define a warping distance, two compared time series trajectories, $T^i, T^j$, are arranged to form a $n^i \times n^j$ grid $G$. The grid cell, $g_{k,l}$, corresponds to the pair ($p_k^i$,$p_l^j$).

**Definition 4.** *A warping path, $W = w_1, \ldots, w_{|W|}$, crosses the grid $G$ such that*

- $w_1 = g_{1,1}$,
- $w_{|W|} = g_{n^i,n^j}$,
- *if $w_k = g_{k_i,k_j}$, then $w_{k+1}$ is equal to $g_{k_i+1,k_j}$, $g_{k_i,k_j+1}$ or $g_{k_i+1,k_j+1}$.*

The order of the locations in a trajectory are maintained but they can be repeated, deleted or replaced by an arbitrary value, a $gap$, along the warping path. The distance is then computed by minimizing or maximizing the sum of a given cost $\delta$ between all pair of locations that make a warping path $W$, for all existing warping paths.

**Definition 5.** *A warping distance is defined as*

$$\begin{aligned} D(T^i,T^j) = &\quad \min_W \left[\sum_{k=1}^{|W|} \delta(w_k)\right], \\ \textit{or} &= \max_W \left[\sum_{k=1}^{|W|} \delta(w_k)\right], \end{aligned} \tag{1}$$

*where $\delta(w_k) = \delta(g_{k_i,k_j}) = \delta(p_{k_i}^i, p_{k_j}^j)$, is the cost function and $W$ is a warping path.*

They are generally computed by dynamic programming. Table III displays the cost functions as well as the dynamic formulation of these distances.

Contrary to the three other distances, $LCSS$ is a similarity. The exact similarity used in Vlachos *et al.*, 2002[14] is $S(T^i,T^j) = \frac{LCSS(T^i,T^j)}{\min\{n^i,n^j\}}$, which is between 0 and 1. We will then use the distance

$$DLCSS(T^i,T^j) = 1 - S(T^i,T^j),$$

to compare distances to each other.

The metric types of these distance functions, and computational cost for the four methods are summarized in table IV.

#### 1) Comparisons:

- All of these distances handle local time shifting.
- The cost function $\delta$ uses the Euclidean distance. Some of these distances have been defined using a L1-norm, but Euclidean distance is more adapted for real values.
- $LCSS$ and $EDR$'s cost function count the number of occurrences where the Euclidean distance between matched location does not match a spatial threshold, $\varepsilon_d$. The former counts similar locations, the latter the difference. This threshold makes the distance robust to noise. However, it has a strong influence on the final results. If the threshold is large, all the distances will be considered similar and if low, only those having very close locations will be considered similar.
- In comparison, $ERP$ and $DTW$ add a weighting to these differences by computing the real distance between

TABLE III: Re-Indexing based distance definition

| | | Cost function $\delta_{NAME}(p_1,p_2)$ = | Distance $NAME(T^i,T^j)$ = |
|---|---|---|---|
| NAME | DTW | $\|p_1p_2\|_2$ | $= \begin{cases} 0 & \text{if } n^i = n^j = 0 \\ \infty & \text{if } n^i = 0 \text{ or } n^j = 0 \\ \delta_{DTW}(p_1^i,p_1^j) + \min\left\{ \begin{array}{l} DTW(rest(T^i),rest(T^j)), \\ DTW(rest(T^i),T^j)), \\ DTW(T^i,rest(T^j)) \end{array} \right\} & \text{otherwise} \end{cases}$ |
| | LCSS | $\begin{cases} 1 & \text{if } \|p_1p_2\|_2 < \varepsilon_d \\ 0 & \text{if } p_1 \text{ or } p_2 \text{ is a gap} \\ 0 & \text{otherwise} \end{cases}$ | $= \begin{cases} 0 & \text{if } n^i = 0 \text{ or } n^j = 0 \\ LCSS(rest(T^i),rest(T^j)) + \delta_{LCSS}(p_1^i,p_1^j) & \text{if } \delta_{LCSS}(p_1^i,p_1^j) = 1 \\ \max\left\{ \begin{array}{lll} LCSS(rest(T^i),T^j)) & + & \delta_{LCSS}(p_1^i,gap), \\ LCSS(T^i,rest(T^j)) & + & \delta_{LCSS}(gap,p_1^j) \end{array} \right\} & \text{otherwise} \end{cases}$ |
| | EDR | $\begin{cases} 0 & \text{if } \|p_1p_2\|_2 < \varepsilon_d \\ 1 & \text{if } p_1 \text{ or } p_2 \text{ is a gap} \\ 1 & \text{otherwise} \end{cases}$ | $= \begin{cases} n^i & \text{if } n^j = 0 \\ n^j & \text{if } n^i = 0 \\ EDR(rest(T^i),rest(T^j)) & \text{if } \delta_{EDR}(p_1^i,p_1^j) = 0 \\ \min\left\{ \begin{array}{lll} EDR(rest(T^i),rest(T^j)) & + & \delta_{EDR}(p_1^i,p_1^j), \\ EDR(rest(T^i),T^j)) & + & \delta_{EDR}(p_1^i,gap), \\ EDR(T^i,rest(T^j)) & + & \delta_{EDR}(gap,p_1^j) \end{array} \right\} & \text{otherwise} \end{cases}$ |
| | ERP | $\begin{cases} \|p_1p_2\|_2 & \text{if } p_1,\ p_2 \text{ are not gaps} \\ \|p_1g\|_2 & \text{if } p_2 \text{ is a gap} \\ \|gp_2\|_2 & \text{if } p_1 \text{ is a gap} \end{cases}$ | $= \begin{cases} \sum_{k=1}^{n^i} \|p_k^i g\|_2 & \text{if } n^j = 0 \\ \sum_{l=1}^{n^j} \|p_l^j g\|_2 & \text{if } n^i = 0 \\ \min\left\{ \begin{array}{lll} ERP(rest(T^i),rest(T^j)) & + & \delta_{ERP}(p_1^i,p_1^j), \\ ERP(rest(T^i),T^j)) & + & \delta_{ERP}(p_1^i,gap), \\ ERP(T^i,rest(T^j)) & + & \delta_{ERP}(gap,p_1^j) \end{array} \right\} & \text{otherwise} \end{cases}$ |

TABLE IV: Re-Indexing based distance properties

| Name | Metric Types | Computation Cost |
|---|---|---|
| DTW | *symmetric* | $O(n^2)$ |
| LCSS | *distance* | $O(n^2)$ |
| EDR | *symmetric* | $O(n^2)$ |
| ERP | *metric* | $O(n^2)$ |

the locations. In this sense they can be viewed as more accurate.

- $ERP$ is the only distance which is a $metric$ regardless of the $L_p$ norm used, yet it works better for normalized sequences, especially for defining the gap value $g$. It does not apply for vehicle trajectories.
- In addition, these distances may include a time threshold, $\varepsilon_t$. Thus, two locations will not be compared if the difference between their time indexing is too large. However, it is very hard to estimate the value of this threshold when comparing trajectories due to the presence of noise.

*2) Pros and Cons:* The main advantage of these distances is that they enable comparison of sequences of different lengths.

The two main limitations of warping based distance are the following

- Warping methods are based on one-to-one comparison between sequences. Hence, it often requires the choice of a particular series that will be used as a reference, onto which all other sequences will be mapped. The index of two sequences being compared should be well balanced in order to best capture the variability, for instance, in order to detect if there were accelerations and decelerations during the measurement of the time series. Hence the choice of the reference sequence is very important.
- The performance of the usual methods based on warping techniques is hampered by the large amount of noise inherent to road traffic data, which is not the case when examining time series.

Instead of correcting the time index, the solution is to use distances that have the effect of time removed.

### *B. Shape-Based Distance*

These distances try to catch geometric features of the trajectories, in particular, their shape. Among **Shape-Based Distances**, the Hausdorff distance (Hausdorff, 1914 [19]), and the Fréchet distance (Fréchet, 1906[20]) are likely the most well known.

*1) Hausdorff:* The *Hausdorff* distance is a *metric*. It measures the distance between two sets of metric spaces. Informally, for every point of set 1, the infimum distance from this point to any other point in set 2 is computed. The supremum of all these distances defines the *Hausdorff* distance.

**Definition 6.** *The Hausdorff distance between two sets of metric spaces is defined as*

$$Haus(X,Y) = \max\{\sup_{x\in X}\inf_{y\in Y}\|xy\|_2, \sup_{y\in Y}\inf_{x\in X}\|xy\|_2\}.$$

This distance is complicated and resource intensive to compute when applied to most existing sets. But in the case of polygonal curves like trajectories, some simplification can be made due to the monotonic properties of a segment. Distance from a point $p$ to a segment $s$ is defined as follows.

**Definition 7.** $Point - to - Segment$ *distance.*

$$D_{ps}(p_{i_1}^1, s_{i_2}^2) = \begin{cases} \|p_{i_1}^1 p_{i_1}^{1proj}\|_2 & \text{if } p_{i_1}^{1proj} \in s_{i_2}^2, \\ \min(\|p_{i_1}^1 p_{i_2}^2\|_2, \|p_{i_1}^1 p_{i_2+1}^2\|_2) & \text{otherwise.} \end{cases}$$

*Where $p_{i_1}^{1proj}$ is the orthogonal projection of $p_{i_1}^1$ on the segment $s_{i_2}^2$.*

Hence, the *Hausdorff* distance between two line segments is

$$\begin{aligned} D_{Haussdorf}(s_{i_1}^1, s_{i_2}^2) &= \max\{ \sup_{p\in s_{i_1}^1} D_{ps}(p, s_{i_2}^2), \\ &\qquad \sup_{p\in s_{i_2}^2} D_{ps}(p, s_{i_1}^1)\} \\ &= \max\{ D_{ps}(p_{i_1}^1, s_{i_2}^2), D_{ps}(p_{i_1+1}^1, s_{i_2}^2), \\ &\qquad D_{ps}(p_{i_2}^2, s_{i_1}^1), D_{ps}(p_{i_2+1}^2, s_{i_1}^1)\}. \end{aligned}$$

Indeed, a segment is monotonic. As seen in Fig. 1, the supremum of the $Point - to - Segments$ distance from any

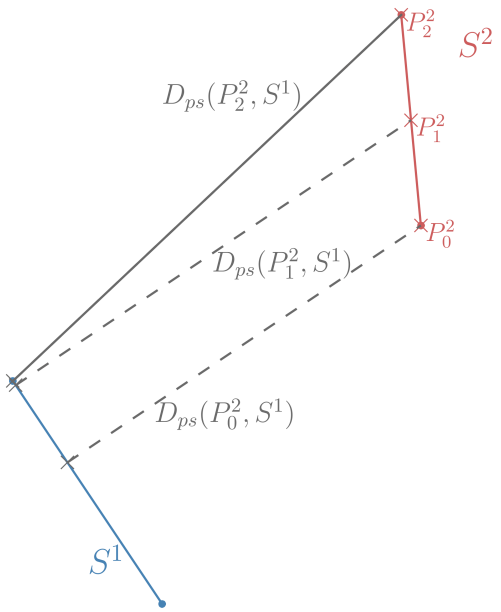


Fig. 1: Supremum of $Point - to - Segment$ distance from point of segment $s_1^1$ to segment $s_1^2$.

point of a segment $s_{i_1}^1$ to a segment $s_{i_2}^2$ occurs at one of the end points of the segment $s_{i_1}^1$. The *Hausdorff* distance between two trajectories can then be computed with the following formula.

**Definition 8.** *Hausdorff distance between two discrete trajectories.*

$$D_{Haussdorf}(T^1,T^2) = \max\{ \max_{\substack{i_1\in[1\dots n^1]\\ j_2\in[1\dots n^2-1]}}\{D_{ps}(p_{i1}^1, s_{j2}^2\}, \max_{\substack{j_1\in[1\dots n^1-1]\\ i_2\in[1\dots n^2]}}\{D_{ps}(p_{i2}^2, s_{j1}^1\}\}.$$

The *Hausdorff* distance can then be computed in a $O(n^2)$ computational time.

*2) Frechet and discrete Fréchet :* The *Frechet* distance measures similarity between curves. It is often known as the "walking-dog distance". Imagine a dog and its owner walking on two separate paths without backtracking from one endpoint to one other. The Fréchet distance is the minimum length of leash required to connect a dog and its owner. While the Hausdorff distance takes distance between arbitrary points, the Fréchet metric takes the form of the two curves into account.

**Definition 9.** *The Fréchet distance between two curves is defined as*

$$D_{Frechet}(A,B) = \inf_{\alpha,\beta\in X}\max_{t\in[0,1]}\left\{\|A(\alpha(t)), B(\beta(t))\|_2\right\}.$$

Similar to the *Hausdorff* distance, the *Frechet* distance is a *metric*. It is also resource intensive. Alt *et al.* (1995[21]) developed an algorithm measuring the exact Fréchet distance for polygonal curves based on the free space definition.

**Definition 10.** *The free space $F_\epsilon(T^1,T^2)$ between two trajectories is the set of all pairs of points whose distance is at most $\epsilon$.*

$$F_\epsilon(T^1,T^2) := \{(p^1,p^2)\in(T^1,T^2)\}|\|p^1,p^2\|_2\le\epsilon\}.$$

The *Fréchet distance* between two trajectories $T^1$ and $T^2$ is the minimum value of $\epsilon$ for which a curve exists within the corresponding $F_\epsilon$ from $(p_0^1,p_0^2)$ to $(p_{n^1}^1,p_{n^2}^2)$ with the property of being monotone existing in both trajectories. Computing the *Fréchet distance* means finding the minimum value of $\epsilon$. By exploiting the monotonic property of the segments and the definition of free space, this task can be accomplished more efficiently.

Indeed, the *Frechet* distance between segments is equal to the *Hausdorff* distance between segments, *i.e.*

$$\begin{aligned} D_{Frechet}(s_{i_1}^1, s_{i_2}^2) &= \max\{ \quad D_{ps}(p_{i_1}^1, s_{i_2}^2),\\ &\qquad\quad D_{ps}(p_{i_1+1}^1, s_{i_2}^2),\\ &\qquad\quad D_{ps}(p_{i_2}^2, s_{i_1}^1),\\ &\qquad\quad D_{ps}(p_{i_2+1}^2, s_{i_1}^1)\}\\ &= \epsilon_{i_1,i_2}. \end{aligned}$$

To compute the *Frechet* distance between trajectories $T^1$ and $T^2$ , we need only look among the set $E$ of *Frechet* distances between all pairs of segments of $T^1$ and $T^2$. $E = \{\epsilon_{i_1,i_2}$ for $(i_1,i_2)\in([1\dots n^1-1]\times[1\dots n^2-1])\}$. This simplification enables us to compute the *Frechet* distance between trajectories $T^1$ and $T^2$ in $O(n^2 log(n^2))$. We highlight that this computational cost is higher than all the other calculation methods of the studied distances.

Eiter *et al.* (1994[22]) describes an approximation of this distance for polygonal curves called the *discrete Fréchet* distance. This distance is close to the definition of the warping based distance.

**Definition 11.** *The discrete Fréchet distance is defined as*

$$D_{Frechet_{Discr}}((T^1,T^2) = \min_W\{\max_{k\in[1\dots|W|]}\|w_k\|_2\}.$$

with $W$ being the warping path defined in definition 5. The *discrete Fréchet* distance can be computed in $O(n^2)$ time.

This distance is bounded as follows.

**Theorem 1.** *For any trajectories $T^i$ and $T^j$ [22]*

$$D_{Frechet}(T^i,T^j) \le D_{Frechet_{Discr}}((T^i,T^j) \le D_{Frechet}(T^i,T^j)+\epsilon$$

*Where,* $\epsilon = \max\{\max_{k\in[1\dots n^i-1]}\{\|p_k^i p_{k+1}^i\|_2\}, \max_{l\in[1\dots n^j-1]}\{\|p_l^j p_{l+1}^j\|_2\}\}$.

*3) One Way Distance:* Lin *et al.* 2005[17] defines the *One-Way-Distance*, *OWD*, from a trajectory $T^i$ to another trajectory $T^j$ as the integral of the distance from points of $T_{pl}^i$ to trajectory $T_{pl}^j$ divided by the length of $T_{pl}^i$

**Definition 12.** *The OWD distance is defined as*

$$D_{OWD}(T^i,T^j) = \frac{1}{n_{pl}^i}\int_{p^i\in T_{pl}^i} D_{point}(p^i T^j)dp^i,$$

where $D_{point}(p,T)$ is the distance from the point $p$ to the trajectory $T$ so that

TABLE V: Shape based distance properties

| Name | Metric Types | Computation Cost |
|---|---|---|
| *Hausdorff* | *metric* | $O(n^2)$ |
| *Frechet* | *metric* | $O(n^2log(n^2))$ |
| *discrete Fréchet* | *symmetric* | $O(n^2)$ |
| *OWD* | *symmetric* | $O(n^2log(n))$ |
| $OWD_{grid}$ | *symmetric* | $O(mn)$ |

$$D_{point}(p,T) = \min_{q \in T_{pl}} \|pq\|_2.$$

The *OWD* distance is not symmetric, but $D_{SOWD}(T^i,T^j) = (D_{OWD}(T^i,T^j) + D_{OWD}(T^j,T^i))/2$ is. This distance is a $symmetric$ because it does not satisfy the triangle inequality.

Lin *et al.*[17], have defined two algorithms to compute the *OWD* in case of piecewise linear trajectories.

- The first consists of finding the parametrized OWD function $D_{OWD}(s_k^i, T^j)$ from a segment $s_k^i$ of $T_{pl}^i$ to all segments $s^j$ of $T_{pl}^j$ and for all segments of $T_{pl}^i$

$$D_{OWD}(T^i,T^j) = \frac{1}{n_{pl}^i} \sum_{k=1}^{n^i-1} D_{OWD}(s_k^i, T^j).\|p_k^i p_{k+1}^i\|,$$

  with a complexity of $O(n^2log(n))$.
- The second uses a grid representation of the trajectory. As we see in Fig. 2, the space is discrete Trajectories are defined as the succession of grids they have crossed.

  **Definition 13.** *A grid representation trajectory is defined as*

$$T_{grid} := (g_0, \ldots, g_{n_{grid}}),$$

  *where $g_n$ are cells of the discrete space.*

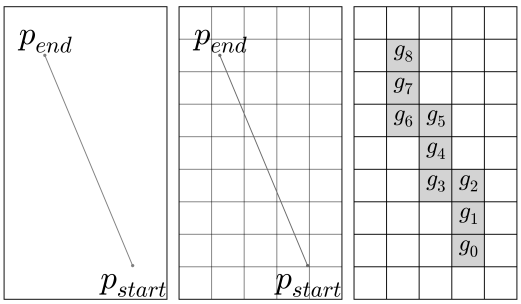


Fig. 2: Grid representation of a segment

This representation simplifies the computation and reduces the complexity to $O(nm)$ where $m$ is the number of local min points. Local min points of a grid cell $g$ are the grids with distances to $g$ shorter than those of their neighbors' grid cell.

Table V displays the metric types and the computational cost of these distances.

*4) Pros and Cons:*

- *Frechet* and *Hausdorff* distances are both *metrics*, meaning they satisfy triangular inequality. With clustering algorithms like *dbscan* or *K-medoid* this is a necessary property of the distance used if we want the clustering algorithm to be efficient. They have been widely used in many domains where shape comparison is needed. But they can fail to compare trajectories as a whole. Indeed both *Fréchet* and *Hausdorff* distance return a maximum distance between two objects at given points within the two objects. As we can see in Fig. 3, despite the fact that the trajectories $T^1$ and $T^2$ are well separated at the maximum value of x, they are clearly more similar to each other than to $T^3$. But with a *Hausdorff* calculated distance, there are no strong differences between $D_{Haussdorf}(T^1,T^2) = 3.26$, $D_{Haussdorf}(T^1,T^3) = 3.02$ and $D_{Haussdorf}(T^2,T^3) = 3.5$. With Frechet, $D_{Frechet}(T^1,T^2) = 6$ is even bigger than both $D_{Frechet}(T^1,T^3) = 4.19$ and $D_{Frechet}(T^2,T^3) = 4.17$.

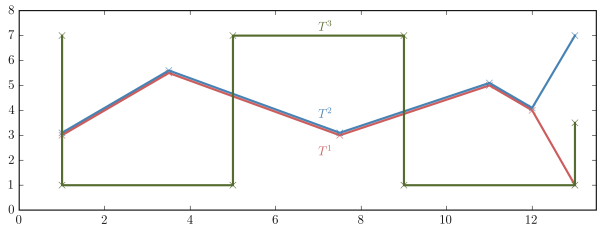


Fig. 3: Frechet And Hausdorff Computation between three trajectories

- The *Discrete Fréchet* distance requires considerably less computing time compared to the *Frechet* distance. But *Discrete Frechet* is not a *metric*. Moreover, due to its similarity with the warping distance it shares the same inconveniences.
- The distance present in Lin *et al.* (2005[17]) is by far the one that best meets our requirements. It compares trajectories as a whole, taking into account their shapes and their physical distances, the required features for our distance. However, its complexity makes it computationally slow. The algorithm for grid representation is faster. Its computational time is $O(mn)$. Yet it does not take into account the computation time required for matching the trajectory to the grid. Moreover, the size of the grid chosen strongly influences the final result and makes it imprecise. Furthermore, the distance gives the same "weight" to all points defining the trajectory: points directly issued from the GPS location, and points which compose the piece wise linear representation. The greater the length of the segment $s$ is, the stronger its influence on the trajectory is. The more separated the endpoints of a segment $s$ are, the less confident the interpolation between them is.

In the following section, a new distance will be established inspired from both the OWD and the Hausdorff distances.

## IV. A NEW DISTANCE : SYMMETRIZED SEGMENT-PATH DISTANCE (SSPD)

In this section, we define a new shape based distance, the *Symmetrized Segment-Path Distance*, and we compare it to other shape based distances. We propose *SSPD* in order to fulfill the desired properties defined in section II-C.

Like the Hausdorff distance (Definition 8), the definition of *SSPD* is based on the $Point - to - Segment$ distance (Definition 7). From this definition, we define the $Point-to-Trajectory$ distance, $D_{pt}$, from a point $p$ to a trajectory $T$ like the minimum of distances between this point and all segments $s$ that compose $T$ (Figure 4). The *Segment-Path distance* from trajectory $T^1$ to trajectory $T^2$ is the mean of all distances from points composing $T^1$ to the trajectory $T^2$ (Figure 5).

**Definition 14.** *$SPD$ distance is defined as*

$$D_{SPD}(T^1, T^2) = \frac{1}{n_1} \sum_{i_1=1}^{n_1} D_{pt}(p_{i_1}^1, T^2).$$

*where,* $D_{pt}(p_{i_1}^1, T^2) = \min_{i_2 \in [0,\dots,n_2-1]} D_{ps}(p_{i_1}^1, s_{i_2}^2)$.

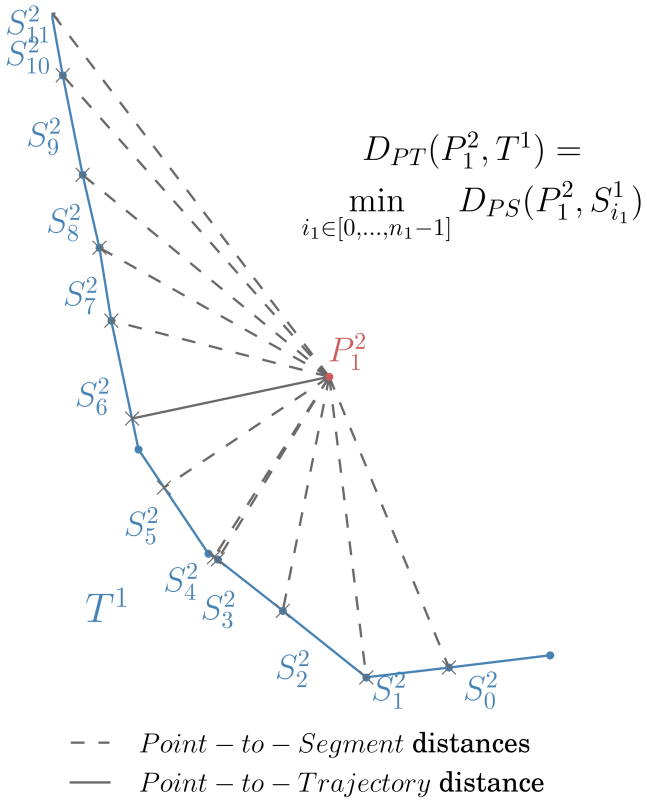


Fig. 4: Distance from point $p_1^2$ to trajectory $T^1$

**Proposition 1.** *If the points that compose the trajectory $T^1$ lie in the set of points $p_{pl}^2$ that compose the piece wise linear representation, $T_{pl}^2$, of trajectory $T^2$, then $D_{SPD}(T^1, T^2) = 0$.*

*Proof.* If the points that compose the trajectory $T^1$ lie in the set $p_{pl}^2$ that compose the piece wise linear representation, $T_{pl}^2$, of trajectory $T^2$, all points of $T^1$ lie within one of the segments, that compose $T_{pl}^2$. By definition $D_{ps}(p_{i_1}^1, s_{i_2}^2) = 0$ $\forall p_{i_1}^1 \in T^1, s_{i_1}^2 \in T_{pl}^2$. It follows that $D_{pt}(p_{i_1}^1, T^2) = 0$ $\forall p_{i_1}^1 \in T^1$ and finally $D_{SPD}(T^1, T^2) = 0$ □

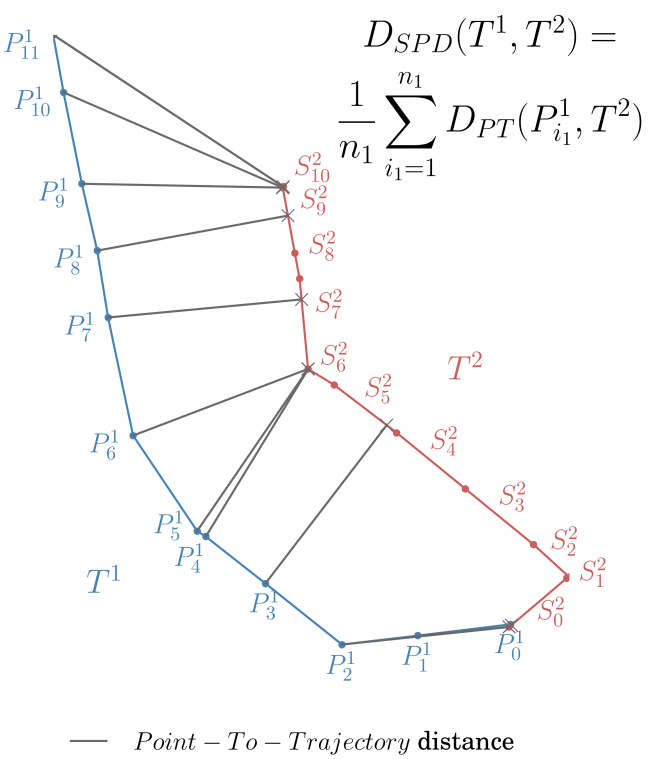


Fig. 5: $SPD$ Distance from trajectory $T^1$ to trajectory $T^2$

This distance is not symmetric. If $T^1$ is a very small sub-trajectory of $T^2$, $D_{SPD}(T^1, T^2) = 0$, $D_{SPD}(T^2, T^1)$ can be very large. By taking the mean of these distances, the **"Symmetrized Segment-Path Distance"**, SSPD, is defined and is symmetric.

**Definition 15.** *Symmetrized Segment-Path Distance*

$$D_{SSPD}(T^1, T^2) = \frac{D_{SPD}(T^1, T^2) + D_{SPD}(T^2, T^1)}{2}.$$

In definitions 14 and 15, distances *SPD* and *SSPD* are computed by taking the mean of the *Point-to-Trajectory* distance and the *SPD* distance. If the maximum is used instead of the mean, one recovers the Hausdorff function between two trajectories. Computing only one distance between two locations makes it very sensitive to noise. Yet our method computes the mean of such quantities which makes it less so. For example, for the trajectories in Fig. 3, the *SSPD* distance between $T^1$ and $T^2$ is smaller than the distance between $T^1$ and $T^3$ or $T^2$ and $T^3$ ($D(T^1, T^2) = 0.58, D(T^1, T^3) = 1.5, D(T^2, T^3) = 2.03$).

**Proposition 2.** *SSDP is a symmetric.*

*Proof.* *SSDP* is a sum of Euclidean distances. By definition *SSDP* is greater or equal to 0. By definition 15, SSDP is symmetric. Finally theorem 1 states that, if $D_{SDP}(T^1, T^2) = 0$, $T^1$ is a sub trajectory of $T^2$. Therefore if $D_{SSDP}(T^1, T^2) = 0$, both $D_{SDP}(T^1, T^2) = 0$ and $D_{SDP}(T^1, T^2) = 0$, and $T^1 = T^2$. *SSDP* is then a *symmetric*. □

*SSDP* is quite similar to *OWD* but its definition resolves most of the problems of *OWD* regarding the desired properties defined in II-C

- The points coming from the interpolation of two observed locations of a trajectory are less trustworthy that the real observations. Hence, it is natural to give more weight to the observed points.
- *SSPD* distance does not require any additional parameters such as a threshold or a grid to be computed.
- Its computation cost is $O(n^2)$. It only depends on the number of locations.

## V. Clustering

To evaluate these different distances, we will study different clustering methods obtained with the same algorithm but with distances computed using all previous distances. The different selected clustering methods and the quality of cluster criterion are examined in this section.

### A. Methods

The choice of the clustering method is restricted by the characteristics of the trajectory object. Indeed, trajectories have different lengths which complicates an easy definition of a mean trajectory object. The *k-means* method cannot be used on our trajectory set, nor *spectral clustering* method. *k-medoid* can be used but an efficient algorithm, like *partitioning around medoids*, or *dbscan* method, require a valid *metrics*. Indeed, these algorithms are based on nearest neighbor and require the distance used to be known in order to satisfy the triangular inequality. Most of the studied distances, *SSPD*, *LCSS*, *DTW*, are not metrics. In this way, *dbscan* or *partitioning around medoids* algorithms will not be used. Moreover, *dbscan* depends on two extra parameters that are hard to estimate in this case.

To perform the clustering of the trajectories, we will focus on two methodologies : *hierarchical cluster analysis* (HCA) and *affinity propagation* (AP). As a matter of fact, *HCA* and *AP* can use distance/similarity which does not satisfy the triangle inequality. We point out that the choice of the clustering method is restricted to the trajectory object we deal with. Actually, trajectories have different lengths. *HCA* and *AP* are both methods which only require the distance/similarity matrix, and thus can cluster objects of different lengths. Both of these methods will be used to evaluate our distance.

### B. Quality criterion of cluster result

A clustering algorithm aims to gather objects into homogeneous groups that are far one from another. Hence, the quality of a clustering is usually evaluated by looking at the between and within variance of the obtained clusters. On the one hand, the within variance shows the spread of elements belonging to the same groups. Because we want the elements of the same groups to be as similar as possible, we want the within variance to be as small as possible. On the other hand, we want objects that belong to different groups to be as far as possible from one another. Hence, the between variance, which shows the spread between clusters, should be as big as possible. The definition of within group variance and between group variance require the definition of a mean object. In our case, they cannot be computed here because of the impossibility of computing the mean of the trajectory object. Therefore, we approximate this mean by considering an exemplar, $T^{ex}$, of a set of a trajectory $\mathcal{T}$ of length $n^{\mathcal{T}}$, defined as:

$$T^{ex}_{\mathcal{T}} = \min_{\substack{T^i \\ i\in[0\dots n^{\mathcal{T}}]}} \Big\{ \sum_{\substack{j=1 \\ j\neq i}}^{n^{\mathcal{T}}} D(T^i, T^j) \Big\}.$$

By using the exemplar definition, we can define new criteria to approximate the between and within variance : the *Between-Like* and the *Within-Like* Criteria. Let $\mathcal{C}_1, \dots, \mathcal{C}_K$ be a set of clusters of $\mathcal{T}$, the *Between-Like* and the *Within-Like* criteria are defined as:

**Definition 16.** *Between-Like and Within-Like*

$$BC = \sum_{k=1}^{K} D(T^{ex}_{\mathcal{T}}, T^{ex}_{\mathcal{C}_k}),$$

$$WC = \sum_{k=1}^{K} \frac{1}{|\mathcal{C}_k|} \sum_{T^i \in C_k} D(T^{ex}_{\mathcal{C}_k}, T^i).$$

The Within-Like criterion shows the spread of elements belonging to the same cluster while the Between-Like criterion shows the spread between clusters. As for the variance, for a given number of clusters, we want the Within-Like criterion to be as small as possible, and the Between-Like criterion to be as big as possible.

These quality criteria can also be used to select the number of clusters. Indeed, the Within-Like criterion decreases with the number of clusters, while the Between-Like criterion increases with the number of clusters. Hence, we are looking for a trade off between these criteria and the number of clusters. We choose the number of clusters so that adding one more cluster does not decrease significantly the Within-Like criteria.

## VI. Experimental evaluation

In this section, we evaluate and compare 7 distances *LCSS*, *DTW*, *Hausdorff*, *Fréchet*, *Discrete Fréchet*, *OWD grid* and the *SSDP*. We present the python package *trajectory_distance* where the distances have been implemented. We compare their computational cost and the results of the application of different clustering techniques for each of them. We also use python for the implementation of the chosen clustering algorithms, the *sklearn* library for *affinity propagation* and *scipy* library for *hierarchical clustering analysis*. For the latter, *weighted*, *average*, *ward*, *complete* and *single* linkage criteria are compared.

### A. A python package : trajectory_distance

All distances have been implemented in python and are available in the *trajectory_distance* package available on github from this url : **https://github.com/bguillouet/traj-dist**.

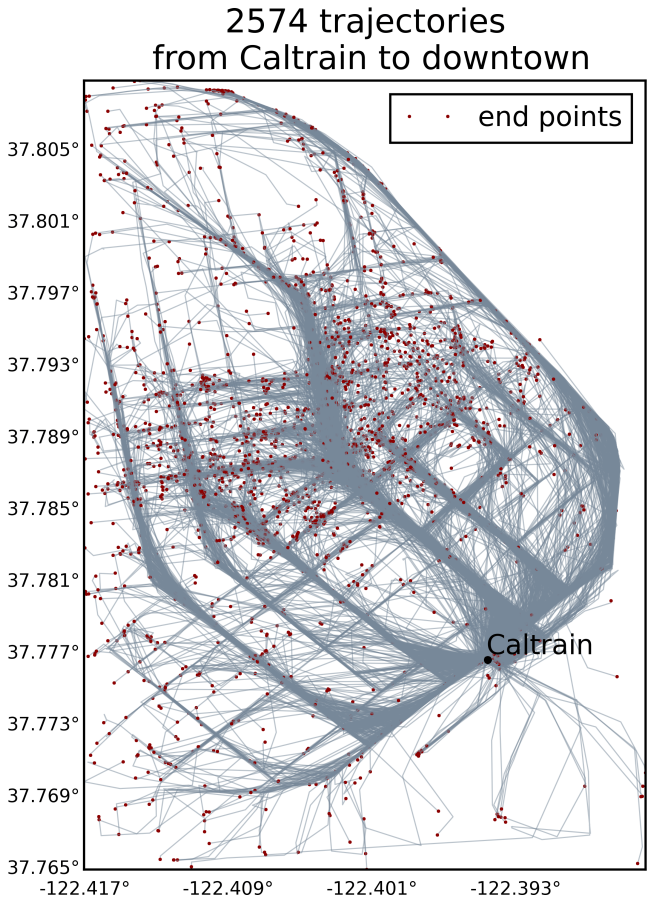


Fig. 6: Trajectories subset

Distances have also been implemented in Cython for 2-D trajectories. Cython is a language which enables the declaration of static variables and to use the C math library. All distances but *OWD grid* are based on Euclidean distance for point to point computation. To compute *OWD grid*, we implemented the grid algorithm defined in [17]. The trajectories need to be mapped in a grid space. We use the Geohash system to accomplish this task. It is based on a hash function which subdivides the geographical space into a grid. Different precision parameters of this grid are available from 1 ($5009.4km \times 4992.6km$ cell) to 12 ($3.7cm \times 1.9cm$). The hash function enables quick mapping of latitude/longitude coordinates to the appropriate grid. Once the trajectory locations have been mapped to a grid, we implemented a algorithm to find all grids they crossed between two locations and to obtain the grid representation of the trajectory as defined in Definition 13.

### B. The Data

The data we used are GPS data from 536 San-Francisco taxis over a 24-day period. These data are public and can be found on [23]. We extracted a subset of this data as shown Fig. 6.

This subset is a blend of 2574 trajectories. All have the same pickup location, the Caltrain station, and all have a drop-off location in downtown San-Francisco.

TABLE VI: Computation Time in seconds

| Distance | Computation time |
|---|---|
| Fréchet | 268.32 |
| Discrete Fréchet | 0.58 |
| Hausdorff | 2.47 |
| DTW | 0.66 |
| LCSS | 0.60 |
| SSPD | 2.46 |
| OWD Grid 5 | 1.88 |
| OWD Grid 6 | 7.44 |
| OWD Grid 7 | 52.96 |

### C. Computation cost

In Table VI we can observe the computation time needed to compute the matrix distance for a subset of 100 taxis trajectories of the studied subset described in section VI-B. Trajectories are composed of 3 to 39 locations, most having around 10.

Fréchet distance is the distance that requires the most computation time. It is the only method that runs in $O(n^2 log(n^2))$. *DTW*, *LCSS* and *Discrete Fréchet* distances are the fastest computed methods, all having the same order of computation time. This three methods require computing the Euclidean distance between each pair of points that compose the two trajectories. They only differ by their cost function. As explained in section IV, *Hausdorff* and *SSPD* distance also are computed in the same way. *Hausdorff* uses the maximum of the *Point-to-Trajectory* distance and *SSPD* the mean, which explains why they have almost the same computation time. If they both have the same complexity as that of *DTW*, *LCSS* and *Discrete Fréchet*, $log(n^2)$, they require more operations, which explains their relative slowness. The time computation of the *OWD grid* is strongly dependent on the precision we choose. Indeed, a precision of 5 resolves the studied space into a $3 \times 3$ grid space, while precisions of 6 and 7 resolve a grid space of $10 \times 5$ and $34 \times 25$ respectively. This implies that the number of cells required to represent the trajectory is very different from one precision to another. With precision 5, 1 to 3 cells are needed to represent the trajectory. With precision 7, the shortest trajectory is represented with only 1 cell but the longest needs 47 cells to fully describe it. Therefore, with precision 5, *OWD grid* is faster than *SSPD*, but with precision 6 and 7, *OWD grid* is 3 times and more than 20 times faster than SSPD distance. A good trade-off needed to be found to have a precision which enabled to adequately describe the trajectories in the new grid space, within a reasonable time computation.

### D. Analysis of the clustering method

In Fig. 7 we observe the evolution of the Within- and the Between- like criteria described in section V for the distance *SSPD* and for the selected methods *AP* and *HCA*. Both the Between-Like and the Within-Like criteria are shown because the sum of these two criteria is not constant as opposed to the sum of the between and within variance.

The *HCA single* method gives much worse results than the other methods, regardless of the number of cluster. All other *HCA* methods display the same evolution of the studied

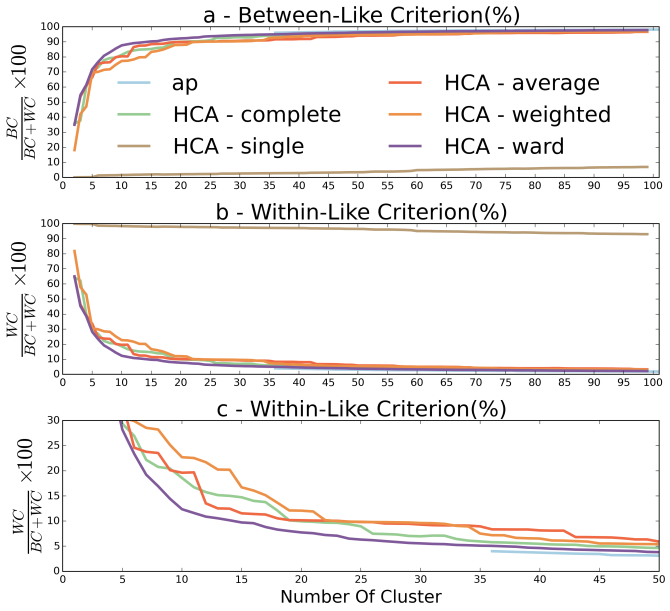


Fig. 7: Evolution of the Between-Like (a) and Within-Like (b) criteria depending on cluster size for clusterings obtained with *SSPD*. On (c), Within-Like criterion is displayed with a reduced scale of 0 to 50.

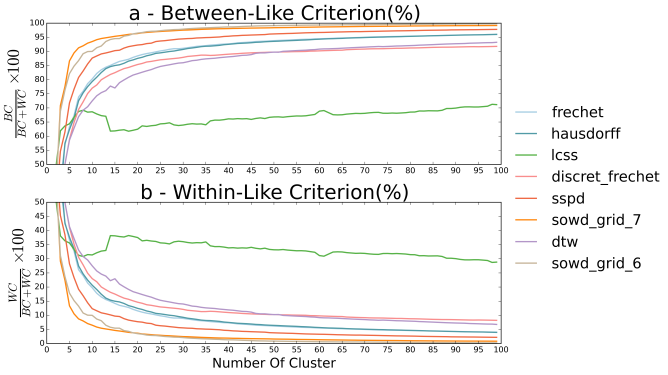


Fig. 8: Evolution of the Between-Like (a) and Within-Like (b) criteria depending on cluster size for all distances using the HCA-WARD method

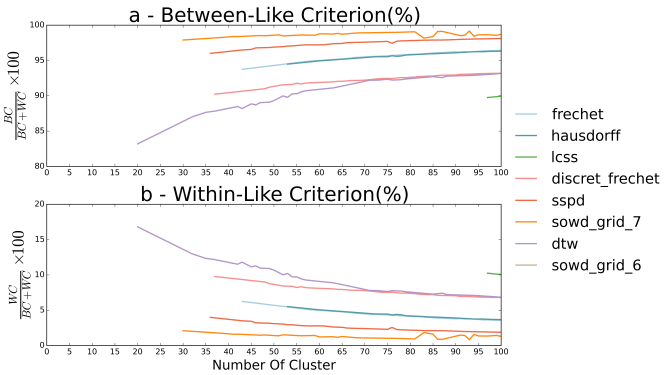


Fig. 9: Evolution of the Between-Like (a) and Within-Like (b) criteria depending on the cluster size for all distances using the AP method

criteria with respect to the cluster size. A plateau can be observed starting at cluster sizes between 10 and 20. Adding more clusters does not decrease significantly the Within-Like Criterion. We can see Figure 7-c, that the *HCA ward* give better results than the other *HCA* method for all the different number of clusters. The same conclusions can be made regardless of the distance used.

*AP* give the best results. However the latter does not achieve clustering using any less than 36 clusters, a potentially large number, given that *HCA* ward achieves clustering around 15, and with good Within-Like criterion results. Moreover, the minimum cluster size found by the *AP* method differs significantly according to the distance used. No less than 21 clusters are found with the *DTW* distance and 54 with *Hausdorff*. This is the main inconvenient of this method.

The *HCA Ward* method and the *AP* method with the *preference* parameter fixed to the minimum of the computed matrix distance will be used to compare the studied distances in more details.

### *E. Analysis of the distances*

We can observe the evolution of the Within-Like and the Between-Like criteria for the two selected clustering methods as well as for all studied distances. The *HCA WARD* results are display in Fig. 8, and the *AP* results in Fig. 9.

In Figure 8, the evolutions of the two criteria when the number of cluster increase is similar for all the distances but *LCSS*. For instance, the curves which represent the Within-Like criterion decrease quickly when adding more cluster for a low number of cluster. Then they all reach a point when adding more clusters does not decrease significantly the Within-Like criterion. Theses point vary from one distance to another. It is reached around 10 clusters for the *OWD grid* and around 25 clusters for the *Discret Frechet* distance. The minimum cluster size found by the *AP* method differs significantly according to the distance used. No less than 21 clusters are found with the *DTW* distance, 36 with *SSPD* and 54 with *Hausdorff*. However, the same conclusion can be made in terms of Within-Like and Between-Like criteria regardless of the number of cluster.

The Warping-based distances, *LCSS* and *DTW*, give the poorest results with *LCSS* being significantly worse than *DTW*. The two shape-based distances *Frechet* and *Hausdorff* give better results. The evolution of their criteria is very similar to one another. The *Discrete Fréchet* distance is between these two types of distances. These results confirm that shape-based distances are better adapted than warping-based distances for our objectives.

*OWD grid* gives the best results. It has the lowest value of Within-Like Criterion for all cluster sizes using both *HCA WARD* and *AP* clustering methods. The results for precision 5 are not displayed here. The discretization of the space is too inaccurate and did not provide good clustering. Precision 7 gives slightly better results than precision 6. This shows that increasing the precision parameter yields better results. However, when there are more than 15 clusters found, the

within and Between-Like criteria are almost the same. In section VI-C we have seen that the computation time to compute the distance with precision 7 is seven times higher than the computation time with precision 6. Hence, we need to look for the optimal criteria to find a good trade off between good clustering results, and reasonable computational time. The choice of this criteria is a strong disadvantage of the *OWD method*, because it implies to look for the best precision parameter for each data set.

Finally, the new distance *SSDP* is the distance which best approaches the results found with *OWD grid*, regardless of the number of cluster. But unlike with *OWD grid*, we do not need to look for the optimal precision parameter, in order to compute it, nor to map the trajectory to a new space. This enables our distance to be more easily adapted to different subsets of trajectories.

We observe the visual results for this distance and both *AP* and *HCA ward* clustering methods, in Fig. 10, and the isolated clusters, in Fig. 11. For the *HCA ward* method, we display the clustering result obtained with 15 clusters because we have seen Section VI-D that a plateau can be observed on the evolution of the Within-Like criteria with respect to the number of cluster starting at cluster sizes between 10 and 20 for the *SSPD*. For the *AP*, clustering results with 36 clusters is displayed since no less cluster can be obtained with this method.

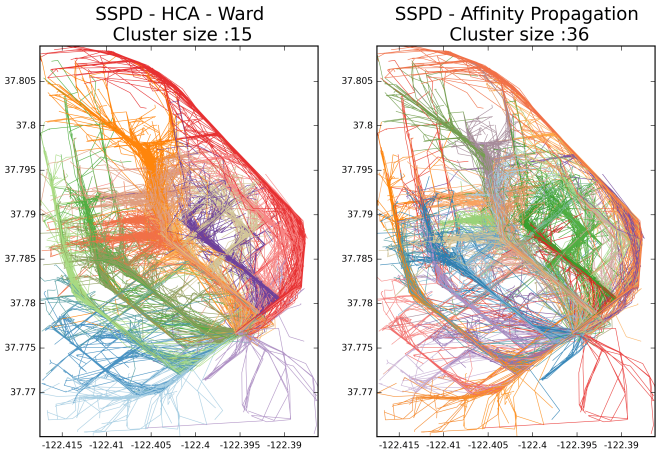


Fig. 10: Clustering results with *SSPD* distance

We observe that trajectories are well classified according to their path. In Fig. 11, clusters found using *HCA WARD* seem to be consistent. The cluster size with *AP* method is 36. This is a large number according to the Within-Like criterion computed with *HCA*. In fact, the Within-Like criterion does not decrease much between 15 and 36. However, we can see that the number of clusters found with *AP* are still consistent.

A cluster computed with the *HCA WARD* method based on a matrix distance computed with *SSPD* gives the best result. The Between-Like and Within-Like criteria show that this method is best used to regroup cluster around exemplar. We obtain a partition of the trajectories subset, such as each cluster represents a path taken by the drivers. We obtain a partition of traffic based on the taxi drivers' behavior leaving the Caltrain

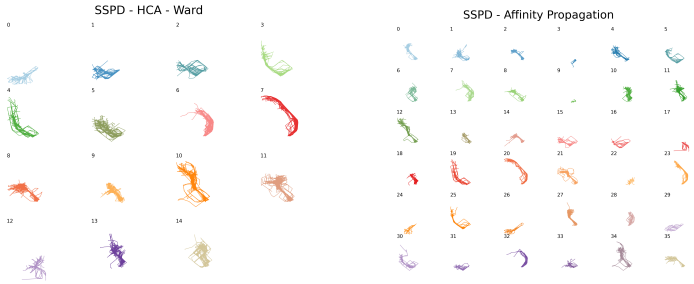


Fig. 11: The isolated clusters

station in San Francisco. The number of trajectories in each cluster gives us a representation of the importance of each network traffic stream.

## VII. Conclusion

Clustering of non Euclidean objects deeply relies on the choice of a proper distance. For trajectories analysis, we presented different distances focusing on different features of such objects. To cope with their different weaknesses we propose a new distance, the *Symmetrized Segment-Path Distance*. This distance is time insensitive, and compares the shape and the physical distance between two trajectory objects. It does not require any additional parameters nor mapping trajectories in a different space. Hence, It can be applied on any set of trajectories, regardless of the area they come from. It enables us to obtain good clustering using either *hierarchical clustering* and *affinity propagation* methods. In this way, the clusters obtained are homogeneous with regard to shape and seem to properly capture the behaviours of the drivers. We have thus obtained a partition of the network based on the drivers' usage that can still be interpreted as vehicle trajectories. This partition can be used to solved different problem. Many applications which recommend places to visit, or which target advertising based on our destination need to forecast the final destination of drivers or predict the travel time of driver trips. Cities which wish to organize trip distribution of a city, also need to know the behaviours of the cars drivers. Some of these problems will be tackled in a following work, based on the partition obtained with our method to cluster trajectories.

## Acknowledgments

We thank the referees for careful reading and numerous suggestions. They led us to an improvement of the work We also thank Frederic and Susan Bejina for helping improve the clarity of the paper.

## References

[1] S. Gaffney and P. Smyth, "Trajectory clustering with mixtures of regression models," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1999, pp. 63–72.

[2] D. Vasquez and T. Fraichard, "Motion prediction for moving objects: a statistical approach," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 4. IEEE, 2004, pp. 3931–3936.

[3] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank, "A system for learning statistical motion patterns," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 9, pp. 1450–1464, 2006.

[4] M. Gariel, A. N. Srivastava, and E. Feron, "Trajectory clustering and an application to airspace monitoring," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 12, no. 4, pp. 1511–1524, 2011.

[5] S. Rinzivillo, D. Pedreschi, M. Nanni, F. Giannotti, N. Andrienko, and G. Andrienko, "Visually driven analysis of movement data by progressive clustering," *Information Visualization*, vol. 7, no. 3-4, pp. 225–239, 2008.

[6] J. Kim and H. S. Mahmassani, "Spatial and temporal characterization of travel patterns in a traffic network using vehicle trajectories," *Transportation Research Part C: Emerging Technologies*, vol. 59, pp. 375–390, 2015.

[7] J.-G. Lee, J. Han, and K.-Y. Whang, "Trajectory clustering: a partition-and-group framework," in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. ACM, 2007, pp. 593–604.

[8] H.-R. Wu, M.-Y. Yeh, and M.-S. Chen, "Profiling moving objects by dividing and clustering trajectories spatiotemporally," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 25, no. 11, pp. 2615–2628, 2013.

[9] E. Tiakas, A. Papadopoulos, A. Nanopoulos, Y. Manolopoulos, D. Stojanovic, and S. Djordjevic-Kajan, "Searching for similar trajectories in spatial networks," *Journal of Systems and Software*, vol. 82, no. 5, pp. 772–788, 2009.

[10] M. K. El Mahrsi and F. Rossi, "Graph-based approaches to clustering network-constrained trajectory data." in *NFMCP*. Springer, 2012, pp. 124–137.

[11] B. Han, L. Liu, and E. Omiecinski, "Road-network aware trajectory clustering: Integrating locality, flow, and density," *Mobile Computing, IEEE Transactions on*, vol. 14, no. 2, pp. 416–429, 2015.

[12] J.-R. Hwang, H.-Y. Kang, and K.-J. Li, *Spatio-temporal similarity analysis between trajectories on road networks*. Springer, 2005.

[13] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series." in *KDD workshop*, vol. 10, no. 16. Seattle, WA, 1994, pp. 359–370.

[14] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *Data Engineering, 2002. Proceedings. 18th International Conference on*. IEEE, 2002, pp. 673–684.

[15] L. Chen and R. Ng, "On the marriage of lp-norms and edit distance," in *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. VLDB Endowment, 2004, pp. 792–803.

[16] L. Chen, M. T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. ACM, 2005, pp. 491–502.

[17] B. Lin and J. Su, "Shapes based trajectory queries for moving objects," in *Proceedings of the 13th annual ACM international workshop on Geographic information systems*. ACM, 2005, pp. 21–30.

[18] M. M. Deza and E. Deza, *Encyclopedia of distances*. Springer, 2009.

[19] F. Hausdorff, "Grundz uge der mengenlehre," 1914.

[20] M. M. Fréchet, "Sur quelques points du calcul fonctionnel," *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, vol. 22, no. 1, pp. 1–72, 1906.

[21] H. Alt and M. Godau, "Computing the fréchet distance between two polygonal curves," *International Journal of Computational Geometry & Applications*, vol. 5, no. 01n02, pp. 75–91, 1995.

[22] T. Eiter and H. Mannila, "Computing discrete fréchet distance," Citeseer, Tech. Rep., 1994.

[23] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "CRAWDAD data set epfl/mobility (v. 2009-02-24)," Downloaded from http://crawdad.org/epfl/mobility/, Feb. 2009.

[24] S.-L. Lee, S.-J. Chun, D.-H. Kim, J.-H. Lee, and C.-W. Chung, "Similarity search for multidimensional data sequences," in *Data Engineering, 2000. Proceedings. 16th International Conference on*. IEEE, 2000, pp. 599–608.

[25] C.-B. Shim and J.-W. Chang, "Similar sub-trajectory retrieval for moving objects in spatio-temporal databases," in *Advances in Databases and Information Systems*. Springer, 2003, pp. 308–322.

[26] A. Srivastava, E. Klassen, S. H. Joshi, and I. H. Jermyn, "Shape analysis of elastic curves in euclidean spaces," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 7, pp. 1415–1428, 2011.


**Philippe Besse** Philippe C. Besse received an Engineering degree in Computer Science from the Polytechnic Institute of Toulouse, France in 1976 and a Ph.D degree in Statistics from the University of Toulouse in 1979. He is currently a full Professor in the Department of Mathematics of the Institut National des Sciences Appliquées of Toulouse, having served as Director for the Departement from 2007-2013. Prior that, he served as the Director of the Laboratory of Statistics and Probabilities, University of Toulouse between 2000 and 2005. He has published more than 50 scientific papers and book chapters in the fields of applied Statistics and Biostatistics. His research interests include functional data analysis, and the industrial applications of Statistics, Bioinformatics and Data Mining.

**Brendan Guillouet** Brendan Guillouet received his Engineering degree in Applied Mathematics from the Institut National des Sciences Appliquées de Toulouse, France in 2013. He is currently doing a CIFRE (Industrial Training and Research) PhD, jointly with Datasio and the Laboratory of Statistics and Probabilities, University of Toulouse. Its thesis focusing on Data Mining and Machine Learning methods applied to mobile data.

**Jean-Michel Loubes** Jean-Michel Loubes received his Phd of Applied Mathematics at University of Toulouse in 2001. CNRS researcher in statistics at University Paris XI and then Montpellier 2, he is since 2007 a full Professor in the Institute of Mathematics of the University, having served as Director for the Departement of Statistics and Probability from 2010-2013. He has published more than 50 scientific papers and book chapters in the fields of applied mathematical Statistics and statistical learning. His research interests include mathematica statistics and the industrial applications of Statistics, and Machine Learning.

**François Royer** François Royer received his Agronomy Engineering degree from the Ecole Nationale Suprieure Agronomique de Rennes, after pursueing PhD studies in marine ecology at CLS from 2003 to 2005, sponsored by the Centre National des Etudes Spatiales and Ifremer. After a two year post-doctoral position at the Large Pelagics Research Lab at University of New Hampshire, conducting field tagging studies and working on astronomical geolocation algorithms, he moved back to the Oceanography Department of CLS in 2007 where he actively worked on underwater geolocation and Argos positioning. He is the author of numerous papers specializing in time series analysis, geolocation filtering and smoothing. He founded Datasio in 2012, a private company focusing on Big Data solutions for industrial and environmental applications, where he heads product innovation and commercial development. His interests range from Bioinformatics and Data Mining to Domain Specific Language development and Functional Programming.